3ISYCP'89
China

A Unifying Approach to the Line Clipping Problem Solution

Vaclav Skala
Computer Science and Informatics Dept.
Institute of Technology
Nejedleho sady 14, P.O.Box 314
306 14 Plzen, Czechoslovakia

Abstract

A new algorithm for 2D line clipping against non convex window that consists of linear edges and arcs is being presented. The general algorithm was derived from the Cohen Sutherland's and Liang-Barsky's algorithms and can be used especially for engineering drafting systems. The algorithm is easy to modify in order to deal with holes, too. The algorithm has been verified on IBM PC in TURBO-PASCAL.

1. Introduction

Clipping is an important part of all graphics packages. There are efficient algorithms as [1],[5],[6] but not for non convex windows that consist of linear edges and arcs. Therefore new algorithms have been developed [8] for clipping 2D line against the general windows. The below described algorithm enables to clip 2D line segments against the windows that are formed by linear edges and circular arcs without need to orient edges in the clockwise or anticlockwise order as some known algorithms require. The algorithm can be easily modified for the hatching. In this case some criteria can be simplified. If the clipping area consists of some holes it is necessary to apply the presented algorithm to all given holes themselves and merge the obtained intersection points together in a convenient way. Particular care was devoted to handle all special situations properly.

2. Non Convex Area Clipping

Provided a non-convex area is given by its vertices in the clockwise or anticlockwise order and if the edge is not linear then information whether the right or left

part of the circle is to be taken from the actual vertex, see fig.2.1. It is also assumed that all vertices have different coordinates, that no vertex lies on an edge or arc and that two edges or arcs might have only a vertex as a common point (for general conditions see [8]). Contrary to the clipping by convex window that consists of linear edges, the line w(q) can intersect the arc edge in two points. It partially increases the complexity of the given problem.

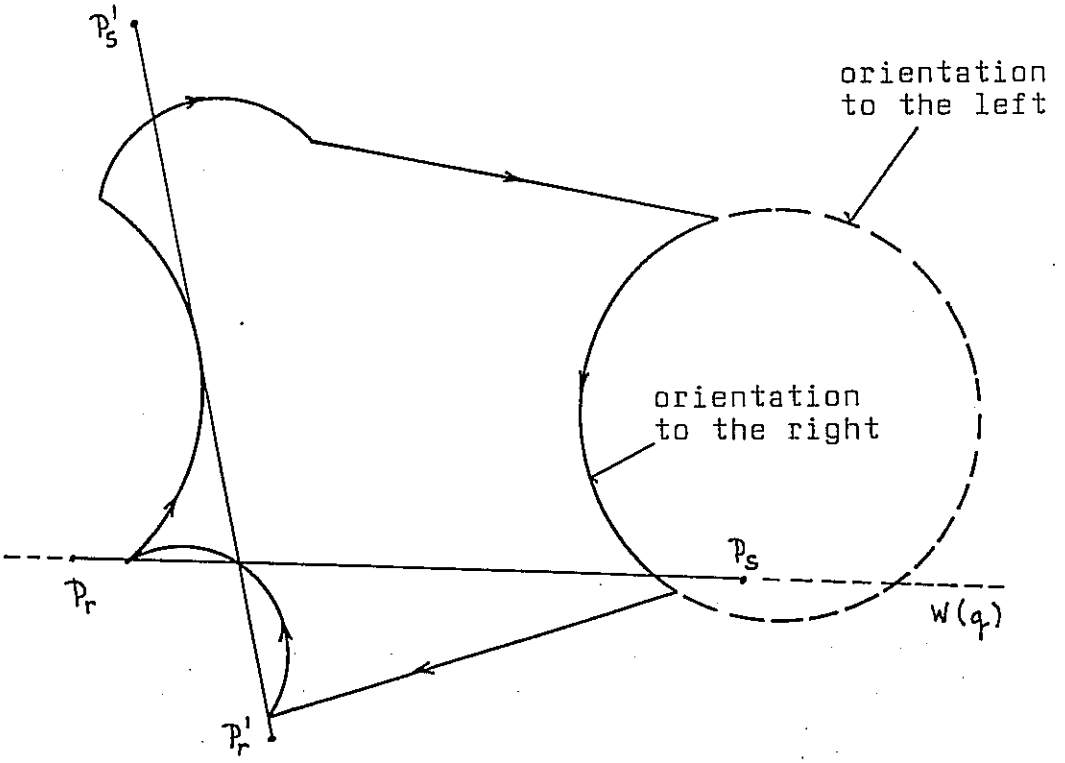


Figure 2.1.

The line w(q) is described by the parametric equation:

$$x(q) = x_r + (x_s - x_r) \cdot q \qquad q \in (-\infty, +\infty)$$

Now in case of arc edge it is necessary to solve the following equations:

$$x(q) = x_r + (x_s - x_r) \cdot q \qquad q \in (-\infty, +\infty)$$

$$(x - x_u)^2 + (y - y_u)^2 - r^2 = 0$$

where (x_u, y_u) is the centre of the given arc
$2r$ is the diameter of this arc.

Solving these equations with regard to variable q the quadratic equation

$$a\,q^2 + b\,q + c = 0$$

will be obtained, where:

$$a = (x_s - x_r)^2 + (y_s - y_r)^2$$

$$b = 2\,[(x_r - x_u) \cdot (x_s - x_r) + (y_r - y_u) \cdot (y_s - y_r)]$$

$$c = (x_r - x_u)^2 + (y_r - y_u)^2 - r^2$$

In the case that the line w(q) intersects or touches the given circle two solutions are obtained, not necessarily different, as:

$$q_{1,2} = (- b \pm \sqrt{b^2 - 4ac}) / (2a)$$

Now it is necessary to determine which part of the circle forms the boundary of the given area. Because the border is oriented it can be discerned whether the arc is on the right or on the left from the connection of $x_K x_{K+1}$ points. If the line w (q) is considered then it must be decided which intersection point ought to be taken. It is obvious that only the point which lies on the proper arc can be considered. It means that:

- if the left arc is considered then the point $x(q_i)$ will be taken into consideration if and only if

$$[s_1 \times s_2]_z > 0 \qquad i=1,2$$

- if the right arc is considered then the point $x(q_i)$ will be taken into consideration if and only if

$$[s_1 \times s_2]_z < 0 \qquad i=1,2$$

assuming that $x_K \neq x(q_i)$, $s_1 = x_{K+1} - x_K$ and $s_2 = x(q_i) - x_K$.

Of course some special situations must be solved again, e.g. when the line passes or touches the vertex x_K In those cases the tangent vectors s_1, s_2, s_3 are determined as:

- for the arc $s_1 = [y_K - y_u , x_u - x_K]$
  where (x_u, y_u) is the centre
  for linear edge $s_1 = [x_K - x_{K-1} , y_K - y_{K-1}]$

- for the arc $s_3 = [\, y_K - y_W \,,\, x_W - x_K \,]$
  where (x_W, y_W) is the centre
  for linear edge $s_3 = [\, x_{K+1} - x_K \,,\, y_{K+1} - y_K \,]$

-for the line w(q) $s_2 = [\, x_S - x_r \,,\, y_S - y_r \,]$

The possible situations are shown in table 2.1.

Table 2.1.

$[s_1 \times s_2]_z$	$[s_3 \times s_2]_z$	type "touch"/"pass"
< 0	< 0	pass
< 0	> 0	touch
> 0	> 0	pass
> 0	< 0	touch
< 0	$= 0$	if $-s_3 . s_2 > 0$ then pass else touch
> 0	$= 0$	if $-s_3 . s_2 > 0$ then touch else pass
$= 0$	< 0	if $-s_1 . s_2 > 0$ then touch else pass
$= 0$	> 0	if $-s_1 . s_2 > 0$ then pass else touch
$= 0$	$= 0$	if $-s_1 . s_2 > 0$ xor $-s_3 . s_2 > 0$ then pass else touch

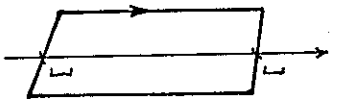
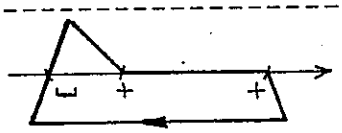
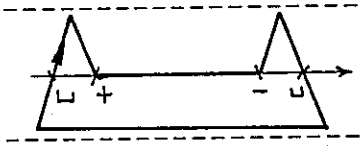
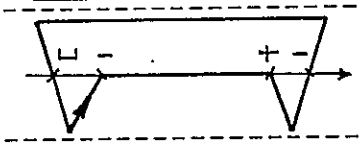
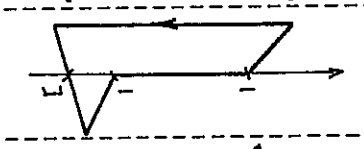
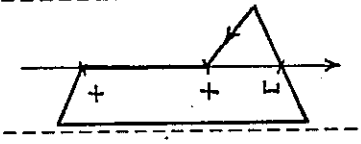
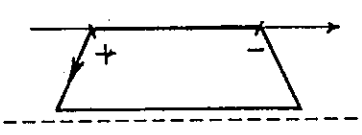
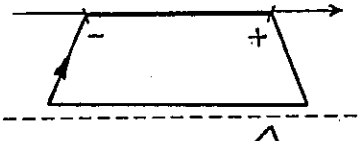
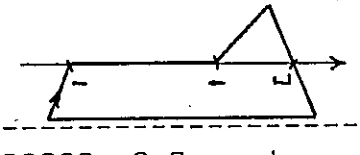
If the arc is oriented to the right then the sign of the tangent vector s must be changed in some situations.

```
PROCEDURE COMPUTE_TANGENT ( xA , xB , r , t );
BEGIN
  IF xA xB is linear
     THEN BEGIN
             s := xB - xA; r := [ s x s2]z;
          END
     ELSE BEGIN
             s := [ yK - yW, xW - xK];
             r := [ s x s2];
             (* (xW,yW) is the centre of the arc *)
             IF r=0 THEN IF t THEN r:=+s.s2 ELSE r:=-s.s2
                    ELSE IF the arc is to the right
                            THEN r := -r
          END
END (* COMPUTE_TANGENT *);

(*                                                        *)

k:=n-1; i:=0;
s2:= xS - xr; (* operation with vectors *)
WHILE i < n DO
BEGIN
  IF xK lies on the line w(q) THEN
```

```
BEGIN
  COMPUTE_TANGENT(x_K, x_i, b, TRUE);
  COMPUTE_TANGENT(x_K-1, x_K, a, FALSE);
  IF x_K x_i is linear THEN
  BEGIN
    COMPUTE_VALUE( q );
    IF a.b > 0
       THEN GENERATE( q , attribute ␣ )
       ELSE IF a.b < 0
               THEN GENERATE( q, q , attribute ␣ )
               ELSE IF a = 0 THEN
                       GENERATE( q , attribute sign b )
                    ELSE
                       GENERATE( q , attribute sign a )
  END
  ELSE (* x_K x_i is the arc *)
  BEGIN
    COMPUTE_VALUES( q_1, q_2);

       IF a.b > 0 THEN
                    GENERATE( q_1, q_2* , attribute ␣ )
       ELSE IF a.b < 0 THEN
                 GENERATE( q_1, q_1, q_2* , attribute ␣ )
            ELSE IF a = 0 THEN
                    GENERATE( q_1, attribute sign b )
                 ELSE GENERATE( q_1, attribute sign a )
  END
  ELSE IF x_K x_i linear THEN
       BEGIN
         COMPUTE_VALUE (q);
         IF an intersection point is inside of (x_K x_i )
         THEN GENERATE ( q with attribute ␣ )
       END
       ELSE
       BEGIN
         COMPUTE_VALUES( q_1, q_2);
         IF an intersection point exists
         THEN GENERATE( q_1*, q_2* , attribute ␣ )
         (* * means if the intersection point lies *)
         (* on the required side of the x_K x_i arc *)
       END
  k := i;
  i := i + 1;
END (* of while *);
SORT ( values q );
REDUCE ( set of q values according to table 2.2. );
SELECT ( subintervals as ⟨q_j, q_j+1⟩∩⟨0 , 1⟩ for all j );
COMPUTE ( the end points );
```

Figure 2.2.

Table 2.2. Basic possible situations for reduction

attributes q_i	q_{i+1}	q_{i+2}	situation	action
␣	␣	*		save (q_i , q_{i+1}); i:=i+2;
␣	+	+		save (q_i , q_{i+2}); i:=i+3;
␣	+	-		save (q_i , q_{i+2}); i:=i+2; change attribute of q to ␣
␣	-	+		save (q_i , q_{i+2}); i:=i+2; change attribute of q to ␣
␣	-	-		save (q_i , q_{i+2}); i:=i+3;
+	+	*		save (q_i , q_{i+1}); i:=i+1; change attribute of q to ␣
+	-	*		save (q_i , q_{i+1}); i:=i+2;
-	+	*		save (q_i , q_{i+1}); i:=i+2;
-	-	*		save (q_i , q_{i+1}); i:=i+1; change attribute of q to ␣

* means all cases, e.g. ␣ + -

The coordinates of the resulted points determined by their q values can be obtained from the equation for the line w(q):

$$x(q) = x_r + (x_s - x_r) \cdot q$$

It is obvious that the presented algorithm for clipping line by non-convex area can be easily modified for a case when the area is formed by linear segments and quadratic arcs. In this case it is necessary to define conveniently the quadratic arcs. A similar approach to the circle case can be chosen for this general case too. Generally all quadratic curves are described by the function $f(x,y)$ together with their tangent vectors as:

$$f(x, y) = 0 \qquad s = [\, f_y, -f_x]$$

where: $$f(x,y) = ax^2 + by^2 + 2cxy + 2dx + 2ey + g$$

If the given area consists of some holes it is necessary to apply the presented algorithm for all the given holes themselves and merge the obtained q values together.

3. CONCLUSION

The presented algorithms are based on the principle of the Liang-Barsky's algorithm. It is shown how the algorithms become more complicated if the requirements are more general. In general they do not need oriented half-planes of the clipping window. The second algorithm solves the situation when the clipping polygon is non convex. The increase of complexity is expressed in the need to distinquish between different cases and to sort the final set of intersection points. The last presented algorithm solves the problem when the clipping area is formed by line segments and arcs. This problem has not been solved in the accessible literature as far as it is known to the author. The algorithms are fast and all special cases are properly handled.

ACKNOWLEDGMENT

The author would like to express his thanks to Prof.L.M.V. Pitteway, Dr.J.P.A.Race (Brunel Univ.,U.K.), Dr.J.E.Bresenham (Winthrop College, U.S.A.) and Dr.R.A.Earnshaw (Univ. of Leeds, U.K.) for their helpful discussions during his stay in U.K., to Miss I. Kolingerova (Inst. of Technology, Plzen) for her interest, comments and the final implementation on IBM-PC, who enabled to find unspecified special cases and errors, and to the students of Computer Graphics course that stimulated this work.

LITERATURE

[1] Cyrus, M., Beck, J., Generalized Two- and Three Dimensional Clipping, Computers & Graphics, Vol.3, 1979, No.1.,pp.23-28.

[2] Foley,J.D.,van Dam A., Fundamentals of Interactive Computer Graphics, Addison-Wesley, 1982, Reading, Mass.

[3] Kilgour,A.C., Unifying Vector and Polygon Algorithms for Scan Conversion and Clipping, Report CSC 87/R7, Computer Science Dept., Univ. of Glasgow, 1987.

[4] Liang,Y.D., Barsky,B.A., An Analysis and Algorithms for Polygon Clipping, CACM 26, No.11 (November), 1984, pp.868-876.

[5] Liang,Y.D., Barsky,B.A., A New Concept and Method for Line Clipping, ACM Transaction on Graphics, Vol.3., No.1.,1984, pp.1-22.

[6] Newman,W.M., Sproull, R.F., Principles of Interactive Computer Graphics, 2nd ed., McGraw Hill, 1979, New York.

[7] Nicholl,T.M., Lee,D.T., Nicholl, R.A., An Efficient New Algorithm for 2D Line Clipping: Its Development and Analysis, ACM Computer Graphics, Vol.21, No.4., 1987, July, pp.253-262.

[8] Skala,V., A Unifying Approach to the Line Clipping Problem Solution TR-209-5-89 Computer Science Dept., Inst. of Technology Plzen, 1989.